NASA CR-
151522

FEASIBILITY STUDY OF MICROPROCESSOR SYSTEMS SUITABLE FOR USE IN DEVELOPING A REAL-TIME PROCESSOR FOR THE 4.75 GHz SCATTEROMETER

(NASA-CR-151522) FEASIBILITY STUDY OF MICROPROCESSOR SYSTEMS SUITABLE FOR USE IN DEVELOPING A REAL-TIME FOR THE 4.75 GHz SCATTEROMETER Final Report, 11 May - 10 Aug. 1977 (Texas A&M Univ.) 56 p

N77-33586

HC A04/MF A01

G3/43

Unclas
50231

Period Covered:
May 11, 1977 — August 10, 1977

supported by

National Aeronautics and Space Administration
Contract NAS 9-15311

TEXAS A&M UNIVERSITY
REMOTE SENSING CENTER
COLLEGE STATION, TEXAS

Final Report RSC 3556-1

FEASIBILITY STUDY OF MICROPROCESSOR SYSTEMS SUITABLE FOR USE IN DEVELOPING A REAL-TIME PROCESSOR FOR THE 4.75 GHz SCATTEROMETER

Period Covered:
May 11, 1977 - August 10, 1977

supported by

National Aeronautics and Space Administration
Contract NAS 9-15311

TABLE OF CONTENTS

Section Page

C-BAND PROCESSOR SYSTEM STUDY

INTRODUCTION

A class of signal processors has been developed at the Remote Sensing Center, Texas A&M University, suitable for the reduction of radar scatterometer data in real-time. The developed systems are applied to the reduction of single polarized 13.3 GHz scatterometer data and provide a real-time output of radar scattering coefficient as a function of incident angle.

It has been proposed that a system for processing of C-band (4.75 GHz) radar data be constructed to support the scatterometer system currently under development. It is the purpose of this report to establish feasible design approaches to the development of this processor system utilizing microprocessor technology.

Two major microprocessor systems, the Motorola M6800 and the Intel 8080, are contrasted and considered for possible use in the development of a processor system. The purchase of a microprocessor system at the chip level is also considered. Recommendations are made as to the most suitable approach in the development of the processor system.

This report is developed into sections considering System Architecture, Trade-Off Considerations, and Recommendations.

SYSTEM ARCHITECTURE

The overall processor system architecture is dictated by the functions it is required to perform. The basic functions are:

1) Sign-sense,
2) Sample power spectra,
3) Obtain aircraft flight parameters,
4) Calculate scattering coefficient,
5) Align coefficients for a single cell, and
6) Output data to recorders.

It was determined from past development that these processing functions can be best accomplished by a hybrid analog/digital processing system. It has also been shown that these functions can be accomplished in real-time under certain processing constraints on cell size and sample frequency.

A system function block diagram for the processor is shown in Figure 1. This system includes analog signal conditioning and sign-sensing, analog spectral sampling, and analog-to-digital conversion for use by the digital microprocessor section.

APPLICATION		REVISIONS			
NEXT ASSY	USED ON	LTR	DESCRIPTION	DATE	APPROVED

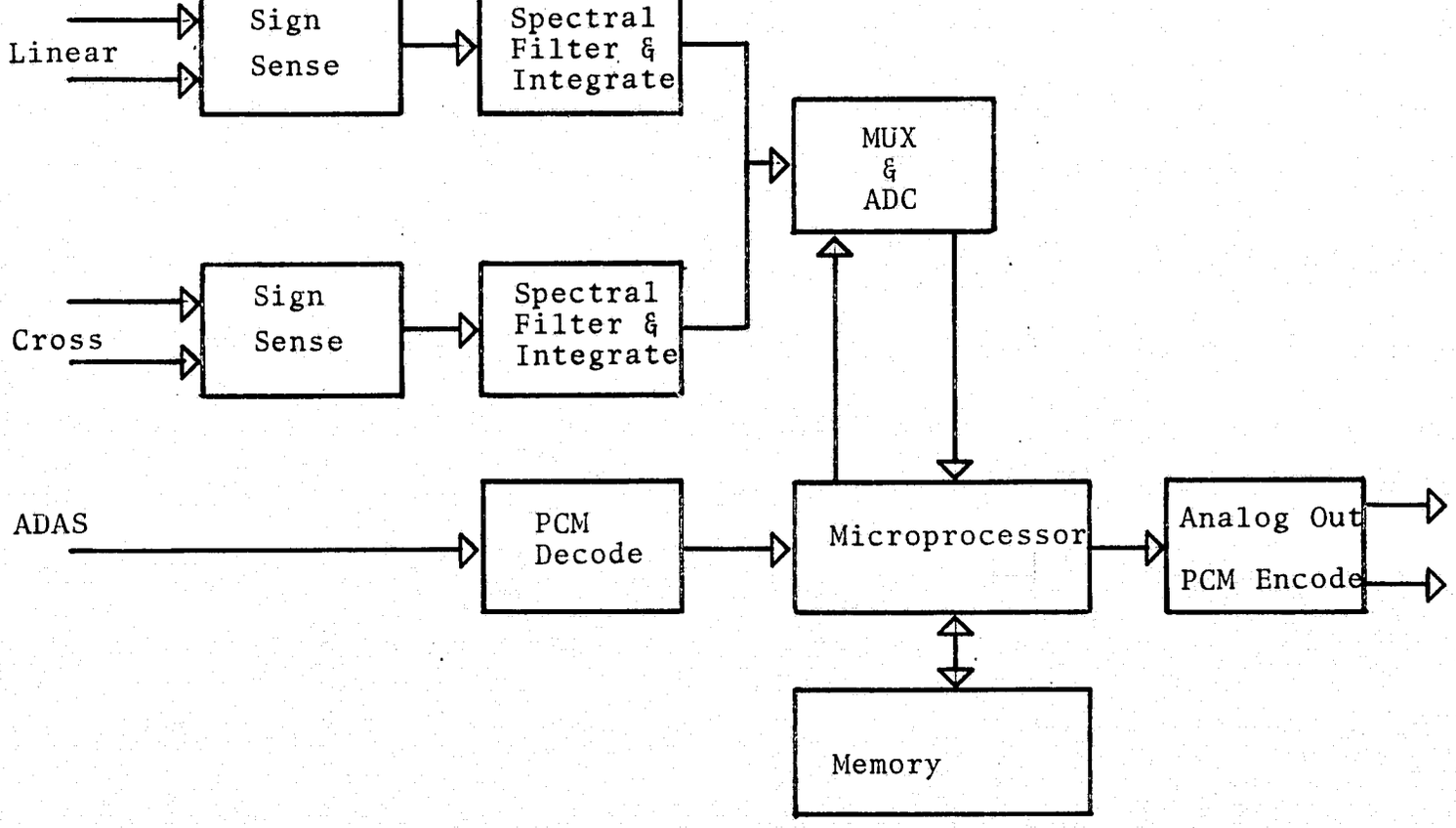


Figure 1
System Function Diagram

TOLERANCES: DEC.± .005 FRACTIONS±1/64 ANGLES±.5°	CONTRACT NO. DATE _____	REMOTE SENSING CENTER TEXAS A&M UNIVERSITY COLLEGE STATION, TEXAS
	ENGR. / CK. / APRVD.	
	SIZE A	CODE IDENT. NO. / DRAWING NO.
	SCALE	SHEET

The digital section consists of a microprocessor, memory and I/O ports. Also included in the system are PCM decoders for input of ADAS aircraft data and PCM encoders for output of processed data to recorders. The fundamental differences considered for the C-band processor are:

1) The type of microprocessor chip,
2) Parallel filtering of spectral data,
3) Dual polarization,
4) Fixed incident angles, and
5) Commercially constructed microprocessor mainframes.

Each subsystem is considered in the following sections.

ANALOG SYSTEM

The Analog subsystem of the processor accomplishes the sign-sense and the Doppler filter functions. Two approaches to implementation of the Doppler filtering in this subsystem are considered.

The first approach is sequential filtering. The Doppler spectrum of the sign-sensed data is sampled by a process which mixes the signal with a voltage controlled oscillator sinusoidal signal. The frequency of this oscillator is selected by the digital processing subsystem so that the desired Doppler information is within a low-pass filter. The filter output is RMS converted, integrated, and analog-to-digital converted for input to the microprocessor system. Multiple Doppler slices are acquired sequentially in a determined pattern to obtain the necessary angle related data. Signals from each of the required angles are measured under the direct control of the system software. Changes in frequencies to select proper angles are readily computed, and changes in the angle selection are software controlled. The two principal parameters in this process are the filter bandpass and the signal integration time.

Doppler filtering can also be accomplished using a parallel filter configuration. Sign-sensed data are filtered by a multiple filter bank where each filter is set

to extract the data from a single scatterometer incident angle. Each signal is RMS converted and integrated. The integrated signals are multiplexed to an analog-to-digital converter which converts the data and provides the results to the digital processor. The principal parameters in this situation are also bandwidth and integration time.

A disadvantage of the parallel filter scheme is that the filter frequencies are fixed, consequently only angles related to these frequencies can be processed. The duplication of filter hardware causes an increase in the construction cost of the parallel filter system over a sequential system, however the implementation is straightforward and maintenance is less complicated.

A relationship which is pertinent to this discussion is the change in incident angle (Θ) for a fixed filter frequency as a function of the change in aircraft velocity (V). This relationship is

$$\Delta\Theta = \tan\Theta \left(\frac{\Delta V}{V}\right)$$

Normal aircraft velocities experiences during data acquisition vary within ± 1.5 knots or about 3%. This converts to a ± 1.5 degrees at a 45 degree incident angle. This is not appreciable for most data gathering situations. Thus, angle variations as a result of different aircraft velocities for different flight lines are small and can be tolera-

ted for the parallel filter configuration.

The number of independent samples of data acquired from a single ground cell is important since the confidence in the estimate of the measurement is related to this number. The normalized standard deviation of the estimated mean (σ/η) is

$$\frac{T}{\eta} = \frac{1}{N} = \frac{1}{\Delta f T}$$

The number of independent samples N is equal to the Bandwidth--integration time product ($\Delta f T$).

Another significant relationship is the length of the ground cell as limited by the Doppler contours corresponding to the bandwidth of the Doppler filter. This relationship is

$$\Delta X = \frac{h\lambda\Delta f}{2v\cos^3\Theta}$$

where h = altitude

v = velocity

λ = wavelength

Table 1 shows the relationship between the independent samples, data variation for worst case standard deviation changes, and ground cell lengths. The table is constructed for an angle of 45 degrees, the filter Q is 100, and integration times are 0.1 seconds for the sequential samples, and 1.0 seconds for parallel samples.

	13.36Hz (20KHz)			C-Band (10KHz)			1.66Hz (5.0KHz)		
	N	L	ΔDb	N	L	ΔDb	N	L	ΔDb
Sequential	20	45m	1.75	10	61m	2.38	5	86m	3.21
Parallel	200	115m	.54	100	130m	.83	50	156m	1.15

Table 1

The best approach to the Doppler filter is not completely evident from the trade-off considerations. Essential considerations are simplicity of design, ground cell resolution and estimation variance.

DIGITAL SYSTEM

Many microprocessors have some of the desirable features which make them attractive for the intended application, but none of them has a complete and consistent set. To be able to evaluate the relative merits of each processor, the advantages provided by each must be examined in detail.

Architecture

Designing a computer architecture can be defined as "determining the needs of the user of the structure and then designing to meet those needs as effectively as possible within economic and technological constraints." [1] Every manufacturer therefore bases the design of his processor on an independently determined set of user needs, and thus produces a processor with a unique if similar implementation.

In the case of the Motorola M6800 and Intel 8080, the similarities are these:

40 pin dual-in-line package

NMOS technology

16-bit 3 state address lines

8-bit 3 state data lines

stack addressing

interrupt capability

Both the 8080 and the 6800 make a stack available to the user. A stack is an area of memory used as a last-in, first-out store. On the 8080 the user may insert on or

remove from the stack any of four 16-bit register pair combinations, while on the 6800 the user may select either 8-bit accumulator, or in the case of interrupts, the condition code, both accumulators, the index register, and the program counter. The 8080 and the 6800 also use the stack for containing the return address when a subroutine is called to provide almost unlimited subroutine nesting.

The 6800 accepts two independent interrupt inputs which cause program execution to be suspended, the current status of the processor to be saved on the stack, and execution to proceed form a specially designated interrupt service routine location. The 8080 has a single interrupt input. The recognition of an interrupt by the 8080 causes program execution to be suspended and execution to continue with the instruction supplied by the interrupting device.

The M6800 and the 8080 A have the following differences:

	M6800	8080A
Instructions	72	78
8-bit registers	2	7
8-bit accumulators	2	1
16-bit registers	2	5
Index registers	1	0
Address modes	8	7
I/O addresses	all addresses	256

	M6800	8080A
Flag bits	6	5
I/O structure	memory mapped	isolated or memory mapped

These statistics are given with the understanding that tables such as this do not completely characterize the differences in the processors because of the simplifications which are made.

Better characterization comes from careful examination of the attributes of each processor. Some of the attributes are:

(1) I/O structure

(2) memory structure

(3) power requirements

(4) operating modes

I/O Structure

The 8080 has two modes for input and output (I/O): isolated and memory mapped I/O. Isolated I/O uses the IN and OUT instructions in conjunction with control information and the decoded peripheral address to transmit to or receive data from an external device. This type of I/O architecture separates the memory address space from the I/O address space and uses a conceptually simple transfer to or from the accumulator. Also, because of the isolation

of program memory and I/O, the full 65K address space is unaffected by I/O addressing. Memory mapped I/O assigns an area of memory space as I/O space. This allows the manipulation of I/O using the same instructions that are used to manipulate memory. Instead of the accumulator as the only transfer medium, any of the internal registers may be used for I/O. It is conceptually more difficult to understand than isolated I/O and limits address space, but it can mean a significant increase in overall speed and a reduction in the required program memory area. The 6800 uses memory mapped I/O for communication with peripheral devices.

Memory Structure

The 6800 addresses memory using the 16-bit address bus and two control signals which are interpreted to determine a valid read or write. Some applications require additional time for addressing. In these cases, the processor may be directed to extend the period during which the address is stable on the address bus for up to 4.5 microseconds. The 8080 addresses memory in the same way, using the 16-bit address bus and two control signals. Operations which require additional address time may inject as many WAIT states as necessary, where a WAIT state is normally 0.5 microseconds long.

Power Requirements

The 8080A requires +5, -5, and +12 volt power supplies. The clocks on the 8080 are not TTL compatible, and must be 8.1 volts in magnitude, assymetrical, and non-overlapping. The 6800 requires a single +5 volt power supply and two phases of TTL compatible clocks which must likewise be assymetrical and non-overlapping. The 6800 and the 8080 read/write memory device requires a single +5 supply. The 6800 and the 8080 serial interface and parallel interface devices both require +5 supplies.

Operating Modes

Both the 6800 and the 8080 are chacterized by their operating modes. For convenience, these may be grouped into three categories:

(1) reset and initialization

(2) execution

(3) interrupt

Item (3) has been discussed above; the remainder of this section will investigate items (1) and (2).

Reset and Initialization

Reset on the 8080 causes the contents of the program counter to be set to zero; thus, at the end of reset, program execution begins at location zero. On

the 6800, reset causes the byte at address FFFE (in hexadecimal) to be loaded into the most significant byte of the program counter and the byte at location FFFE to be loaded into the least significant byte of the program counter. Thus, at the end of reset, program execution begins at the location contained in the 16-bit number beginning at FFFE. It can be seen that while the RESET function on the 8080 and the 6800 is implemented differently, they may be functionally equivalent.

Execution

Both the 6800 and the 8080 execute a program by repetitively fetching instructions from memory, then controlling the internal and external circuits in accordance with the information coded into the respective instructions. Instructions may consist of one, two, or three bytes of information, and are fetched from the memory addressed by the current contents of the program counter (PC). The first byte contains sufficient information to define how many additional bytes are required as well as which operation is to be performed. The 6800 has a 1 MHz clock rate yielding an effective data transfer rate of 1 Mbit/sec., while the 8080 has a 2 MHz clock with an effective data transfer rate of 840 kbit/sec. [2].

SOFTWARE

Differences in the characteristics of the 6800 and 8080 software which affect the performance of the required processing are

(1) variation of coding and execution efficiencies for typical operations

(2) lack of conditional absolute branch instructions and conditional subroutine call and return instructions on the 6800

(3) variation in subroutine linkage conventions

(4) variation in typical subroutine design

Variation of Efficiency

A typical arithmetic operation used in the flight program is a 16-bit addition. The implementation for the 8080 and the 6800 is given below:

8080:

bytes	time (μs)	instruction
3	8	LHLD ØPI
1	2	XCHG
3	8	LHLD ØP2
1	5	DAD D
3	8	SHLD RES
11 bytes	31 μs	

6800:

bytes	time (μs)	instruction
3	4	LDAA ØP1+1
3	4	ADDA ØP2+1
3	5	STAA RES+1
3	4	LDAB ØP1
3	4	ADCB ØP2
3	5	STAB RES
18 bytes	26 μs	

These codings assume that the operands are in memory and that the results are stored in memory. The 8080 is assumed to have a characteristic 2MHz clock rate and the 6800 a characteristic 1MHz clock rate. The memory efficiency of the 8080 is 39% greater than 6800, while the execution efficiency at the 6800 is 16% greater than the 8080. The memory efficiency of the 8080 is due to its 16-bit load and store instructions, and the execution efficiency of the 6800 is due to its capability to add to an accumulator from memory.

If the assumption is made that one operand is already in register pair D in the 8080 or in the accumulators in the 6800 (which is typically the case), the programs above become:

8080:

bytes	time	instruction
3	8	LHLD ϕP2
1	5	DAD D
3	8	SHLD RES
7 bytes	21 μs	

6800:

bytes	time	instruction
3	4	ADAA ϕp2+1
3	5	STAA RES+1
3	4	ACDB ϕP2
3	5	STAB RES
12 bytes	18 μs	

In this case, the coding efficiency of the 8080 is 42% greater than that of the 6800, while the execution efficiency of the 6800 is 14% greater than the 8080.

Conditional Control Transfer

The 8080 provides 8 conditional jump instructions. These execute in 5 μs and require 3 bytes of memory for the instruction and a 16-bit branch address. The 6800 provides 14 conditional jump instructions, each of which use the relative addressing mode and require 2 bytes and execute in 2 μs. However, if a conditional branch is required to an address outside of program counter relative range, the operation must be recoded as

a relative branch and an absolute jump, requiring 5 bytes and 11 μs. Studies have shown that 80% of all conditional branch addresses lie within relative addressing range [1], so the net difference due to conditional jumps in overall execution time between the 8080 and the 6800 is insignificant.

The 8080 also has conditional subroutine call and return instructions. The 6800 does not. Again, these must be coded as a relative branch and a call or return, requiring 5 bytes and 11 μs for a call and 3 bytes and 7 μs for return. The 8080 performs an equivalent operation in 3 bytes and 8.5 μs for a call and 1 byte and 5.5 μs for a return.

Subroutine Linkage

When a subroutine is called on the 8080, the arguments are passed in the registers. The 6800 has too few registers to use this convention, and an alternate method must be used. One method is to use an area of memory to contain arguments and results. The relative efficiencies of the two methods are compared below:

8080:

bytes	time (μs)	instruction
3	8	LHD ØP1
1	2	XCHG

3	8	LHLD ØP2
1	2.5	MØV B,H
1	2.5	MØV C,L
9 bytes	23 µs	

6800:

bytes	time (µs)	instruction
3	5	LDX ØP1
3	6	STX SAVE1
3	5	LDX ØP2
3	6	STX SAVE2
12 bytes	22 µs	

The memory efficiency of the 8080 is 75% greater than that of the 6800 while the execution efficiency of the 6800 is 4% greater than the 8080.

Subroutine Design

In the 8080 mathematics subroutines, the H register pair is used as an index into the tables, making computation of the effective table address a single DAD. The 6800 does not allow a 16-bit add, so for an equivalent operation the look-up procedure must be recorded. Representative codings for the 8080 and the 6800 are given below:

8080:

bytes	time	
3	5	LXI H,TAB
2	3.5	MVI B,O

1	2	ADD	A
3	5	JNC	NC
1	2.5	INR	B
1	2.5 NC	MØV	C,A
1	5	DAD	B
1	3.5	MØV	E,M
1	2.5	INX	H
1	3.5	MØV	D,M
15 bytes	35 µs		

6800:

bytes	time		
3	4	LDAB	TAB+1
1	2	ABA	
3	4	LDAB	TAB
2	4	BCC	NC
1	2	INCB	
3	5 NC	STAA	SAVE3
3	5	STAB	SAVE3+1
3	5	LDX	SAVE3
2	5	LDAA	0,X
2	5	LDAB	1,X
3	5	STAA	SAVE3
3	5	STAB	SAVE3+1
29 bytes	51 µs		

8-Bit Arithmetic and Logic Functions

The 8080 and the 6800 have equivalent 8-bit arithmetic and logic functions. In addition, the 6800 allows arithmetic shifts which the 8080 does not. The 6800 also has an overflow flag bit to detect two's complement overflow.

Both the 6800 and the 8080 require one byte and 2 μs for the execution of a register-to-register operation.

The 8080 register-to-memory operations take one byte (two using immediate addressing) and execute in 3.5 μs. The 6800 register-to-memory operations take three bytes (two using immediate addressing) and execute in 2 to 4 μs.

The 6800 rotate instructions are 9-bit (including carry) and may operate on either accumulator or on memory. The 8080 has 8- and 9-bit rotates for the accumulator only.

The 6800 allows arithmetic shifts for the accumulators and memory. In a right shift, the MSB (sign) big is propagated. In a left shift, zero is propagaged into the LSB.

Execution time of both 6800 and 8080 register shifts and rotates is 2 μs. A 6800 memory shift or rotate requires 6 to 7 μs.

Memory Size and Usage

The 8080, with its separate I/O systems allows the full 64K memory addressing range to be used or program storage. The 6800, with its memory mapped architecture, must split its range between I/O and program storage. The 6800 tends to use more memory for programs, and more memory accessing takes place, due to the lack of program registers.

The 8080's multi-register architecture allows more usage of register-to-register instructions. More memory may be devoted to EPROM because of this occurrence. Since the 6800 must use more register-to-memory instructions, more memory must be devoted to RAM.

A tabulation of the instructions used in the 8080 implementation of the program reveals that the instruction CALL was used 390 times. If only 50% of these involve argument setup as described above, then the 6800 requires 260 more bytes than the 8080. Also, the 8080 uses 148 return instructions of various types. If half of these involve subroutines with arguments, then the 6800 requires 1036 more bytes than the 8080, making the additional memory required about 1300 bytes over the 8080. Coupled with this fact is a time cost overhead of 16 additional microseconds with every subroutine call. If all of the subroutines with arguments were called only once, then the 6800 would re-

quire an additional 1.2 milliseconds of time simply to invoke the subroutines. In reality, subroutines are called many more times than they are coded in the program, making the actual time cost of the 6800 much higher than the 8080.

Speed

The 6800, although it uses a slower cycle time, has an execution speed that is approximately the same as the 8080. The 8080 uses more machine cycles for its operations, in spite of its register-to-register capabilities.

ADDRESSING MODES

Extended Addressing

The Extended Addressing mode permits the MPU to refer to any memory location within the 65,536 bytes which the MPU can directly access. Instructions using the extended addressing mode consists of three bytes, the first of which is the instruction operation code. The second byte contains the most significant eight bits of the operand address and the third contains the least significant eight bits of the operand address.

The operation code for instructions with the extended addressing mode has the form, rr11xxxx, where rr = 00. In other words, the hexadecimal representation of the operation code has the first digit of 7, B, or F. The following instructions may be used with the extended addressing mode:

ADC,ADD,AND,ASL,ASR,BIT,CLR,CMP,COM,
CPX,DEC,EOR,INC,JMP,JSR,LDA,LDS,LDX,LSR,
NEG,ORA,ROL,ROR,SBC,STA,STS,STX,SUB,TST

Direct Addressing

The Direct addressing mode permits the MPU to access any memory location within the first 256 bytes, which is also known as "Page0." Instructions using the Direct addressing mode consist of two bytes, the second

byte of which is the location on page 0, or the least significant eight bits of the operand address; the most significant eight bits of the operand address is implicitly zero.

The operation code for the instructions which use the direct addressing mode has the form, 1r01xxxx, where r designates the second operand (A or B,s or X). The hexadecimal representation of the operation code has the first digit of 9 or D.

The following instructions may be used with the direct addressing mode:

ADC,ADD,AND,BIT,CMP,CPX,EOR,LDA,LDS,
LDX,ORA,SBC,STA,STS,STX,SUB

Indexed Addressing

The Indexed addressing mode permits the MPU to access a memory location relative to the contents of a 16-bit index register. This is actually a "base displacement" form of addressing, in that a one byte offset is added to the value in the index register to form the true memory address. The index register is not altered by this addressing mode. Instructions using the indexed addressing mode consist of two bytes, the second of which defines an offset or displacement of between zero and 255, inclusive.

The operation code for instruction using the indexed addressing mode has the form, rr10xxxx, rr = 00. The hexadecimal representation of the operation code has the first digit of 6, A, or E. The following instructions may be used with indexed addressing mode:

ADC,ADD,AND,ASL,ASR,BIT,CLR,CMP,COM,
CPX,DEC,EOR,INC,JMP,JSR,LDA,LDS,LDXLSR,
NET,ORA,ROL,ROR,SBC,STA,STS,STX,SUB,TST

When the indexed addressing mode is used with instructions which operate on the index register (LDX,STX, CPX), the address is formed from the initial contents of the index register, and is not affected by any changes to the index register during the course of the instruction execution.

Relative Addressing

The Relative Addressing mode permits the MPU to access a memory location relative to the current contents of the Program Counter. This addressing mode is used for the branch instructions only, and is the only addressing mode used for the branch instructions, all of which are two bytes in length. The effective address, which becomes the location of the next instruction (by being loaded into the program counter), consists of the algebraic sum of the contents of the program counter (containing

the address of the instruction following the branch) and the second byte of the instruction, treated as a signed, two's complement number in the range +127, -128. Thus the MPU can branch forward to an instruction 129 locations from the beginning of the branch instruction, backward to an instruction 126 locations from the beginning of the branch instruction, or anywhere in between, for a total range of 256 locations.

The operand code for the branch instructions is either of the form, 00010xxxx, or of the form, 10001101. That is, either hexadecimal 8D or a first digit in hexadecimal of 2. The following instructions use the relative addressing mode:

BCC,BCS,BEQ,BGE,BGT,BHI,BLE,BLE,BLT

BMI,BNE,BPL,BRA,BSR,BVC,BVS

Immediate Addressing

The Immediate Addressing mode permits the operand of an instruction to be included with the instruction itself. Most of these relate to eight-bit registers. and require one-byte operands, making the immediate-addressed instruction two bytes long. Three instructions, however, relate to one of the two 16-bit registers (Index register or Stack pointer), so that two-byte operands are required; these

are LDS, LDS, and CPX, and require three bytes in the immediate addressing mode. The first byte in either case is the operation code; the remaining one or two bytes are the operand data. If there are two bytes of operand, the first of the two is the most significant eight bits, and the second is the least significant eight bits.

The operand code for the instructions using the immediate addressing mode has the form, 1r00xxxx, where r selects the second operand (destination of the data). The following instructions may be used with the immediate addressing mode:

ADC,ADD,AND,BIT,CMP,CPX,EOR,LDA,LDS,
LDX,ORA,SBC,SUB

Inherent Addressing

All of the instructions which are not specifically defined to have a memory addressing mode defined above are considered to have the Inherent Addressing mode, and consist of one byte only, with the addressing mode inherent in the operation code byte. These may be subdivided into the following categories: No Operand, Stack Addressing, and Register Addressing, which may itself be subdivided into Accumulator Reference, and Other Register Reference. There is no consistent pattern for the operation codes of the instructions which use the inherent

addressing mode, except that they all have the first hexadecimal digit of 0,1,3,4, or 5.

Comparisons

1. The 8080 and 6800 Immediate modes are equivalent.
2. The 8080 Direct and 6800 Extended modes are equivalent.
3. The 8080 Register Indirect is similar to the 6800 Indexed mode in that an "index register" is used, but no relative offset is available. The 8080 has 3 register pairs to be used (BC,DE,HL), while the 6800 has one Index Register.
4. The 8080 has no "page zero" addressing.
5. The 8080 has no relative addressing for either operand fetch or branching. The relative addressing feature is useful for relocation of code segments. The size of a program also decreased.

UTILIZATION OF COMMERCIALLY AVAILABLE MAINFRAMES

Both processing families have been found to contain four levels of system complexity.

Chip level systems and the support IC's supplied with them were the first level introduced for market penetration. Generally unusable at this level, the chip kits require extensive additional interface support to bring them up to a operational plane.

Level two is now defined as the card level microprocessor. The card always contains a small amount of scratch pad with positions supplied for E-PROM or ROM. RS-232-C and/or teletype interface is supplied for entry.

The card level system is usually supplied with a small package of operating software which aids the user in program construction and debug. Physical interface is made through card edge connectors and/or flat cable which facilitates access to address and data buss along with the existing interrupt and control flags. Utilization of the card is accomplished through interface with keyboard or CRT, after having supplied chassis, card-cage, power supplies, interface cables and the support software. Based on the system procured, additional card level components are available as options.

The next step up, or level three, can be defined as the basic operational processor.

The fundamental processor card is supported by all the required physical interface. Chassis, card cage, power, cables and a small amount of memory are supplied with any number of add-ons or options available. With the addition of an entry device, it is at this level that the systems become attractive for immediate use.

Level four steps into the micro/mini system stage. Large software packages are available along with finished mainframes and operator panels. Some of the systems at this level included CRT terminals, keyboards or at least an operator switch register for conversation with the CPU.

Price ranges for these various levels may be defined by:

1.	Chip level system	$\simeq$ \$100-200
2.	Card level system	$\simeq$ \$500-1000
3.	First level mainframe	$\simeq$ \$2500-4000
4.	Second level mainframe (mini)	$\simeq$ \$4000-10000

In this particular application, level three, or the first level mainframe, is thought to be the most applicable. Investigation has shown that both families contain the required hardware, software, and expansion capabilities to facilitate the additional NASA special purpose components. Price, manufacturing, reliability, maintainability and the other tradeoffs required, all appear to be equitable at this level.

THE TRADEOFF WITH AND WITHOUT UTILIZING A FIRST LEVEL MAINFRAME

The primary areas of savings when employing a mainframe system are in time and commonality. Assuming that a first level mainframe is procured, savings are predicted in the areas of:

1. Mainframe, design and fabrication
2. Motherboard design and fabrication
3. Interconnect design and fabrication
4. CPU design and fabrication
5. Memory design and fabrication
6. Power source design and fabrication
7. Interface design and fabrication
8. System integration
9. Lead time
10. Component commonality with existing systems

The parts procured at this level are generally produced to a "best commercial practices" specification. Flexibility of design, reliability, and quality all are approximately equal family-to-family and although not military-grade, possess the potential for being upgraded to engineering model flight worthiness. Costs involved in the tradeoff when considering board level components as listed above, are usually less for the procured subassemblies because of fabrication and checkout time for inhouse boards run two or three to one.

Investigation has indicated that the "make or buy" breakpoint for cost exists above the first level mainframe. Siginificant savings are experienced in the areas of common mechanical and electronic fabrication, lead-time, debug, commonality, and interface, while the system does not require the general purpose features of the second level, or micro/mini system.

CONCLUSIONS

The following conclusions are made from this study:

1. Intel 8080 and Motorola M6800 microprocessors are for all practical purposes equivalent in their present hardware configuration, speed, and reliability.

2. The 8080 series has an architecture which lends itself to further development and projections point to more sophisticated 16-bit microprocessors which are compatable to the 8080 architecture.

3. The M6800 architecture limits the further development of this series of microprocessor and probably is the last major development in the series.

4. Development software support for the Intel 8080 significantly exceeds that of any other available microprocessor and vendor support in this area is excellent.

5. Commercially available microprocessor systems are adequate for the development of processor systems. Special purpose subsystems can be added in a straightforward fashion using vendor built mechanical hardware.

6. Parallel analog filtering represents a straightforward approach to Doppler filtering of scatterometer signals. However, it is more expensive to construct and has no angle selection flexibility. This method exhibits a greater noise immunity than a sequential filter approach with a consequent decrease in ground cell resolution.

RECOMMENDATIONS

As a result of this study, recommendations based on the stated conclusions are:

1) An Intel 8080 series microprocessor be used to implement the C-band processor system,

2) A first level Intel mainframe be purchased for the implementation, and that additional functions required for processing be added to the mainframe,

3) A second-mainframe chassis be purchased to house the analog portion of the processor, and

4) That a parallel approach to Doppler filtering be used rather than the sequential approach.

REFERENCES

[1] F.P. Brooks, Jr., "Architectural Philosophy," in Planning a Computer System-Project STRETCH, W. Buchholz, ed., McGraw-Hill, 1962, p.5.

[2] Bala Parasuraman, "High-Performance Microprocessor Architectures," Proc. IEEE, Vol. 64, No. 6, June 1976, p.852.

APPENDIX

Presentation of Results of Trade-off Study for a C-Band Scatterometer Processor

Johnson Space Center

June 10, 1977

STUDY OBJECTIVES

I.

AN ENGINEERING TRADE-OFF OF AVAILABLE MICROPROCESSORS. (PRIMARILY THE INTEL 8080 AND MOTOROLA 6800.)

II.

AN INVESTIGATIVE TRADE-OFF OF THE COMMERCIALLY AVAILABLE MAINFRAMES ALONG WITH PREVIOUS FAB METHODS.

III.

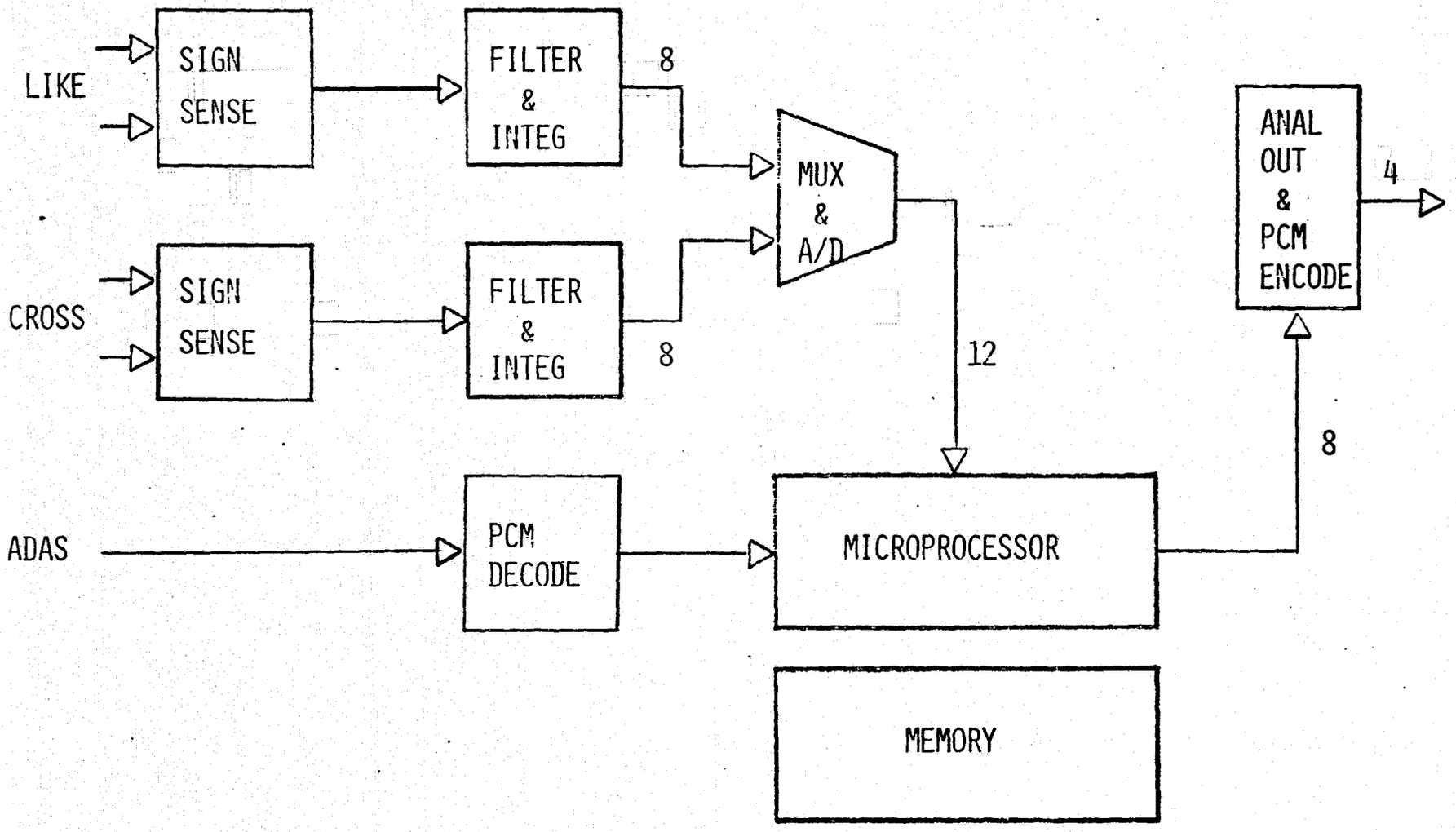
CONSIDERATION OF PARALLEL VERSUS SEQUENTIAL DOPPLER FILTERING APPROACHES.

IV.

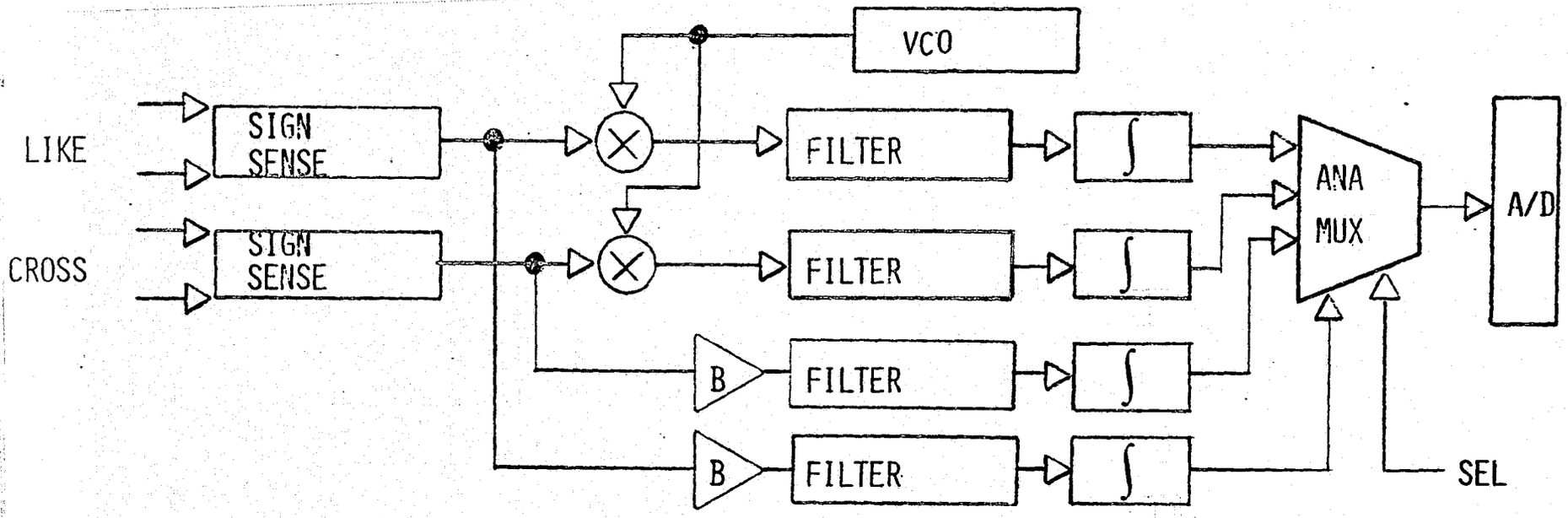
INTEGRATE RESULTS INTO A RECOMMENDATION FOR THE 4.75 GHz SCAT.

A PROCESSOR MUST FUNCTIONALLY PERFORM:

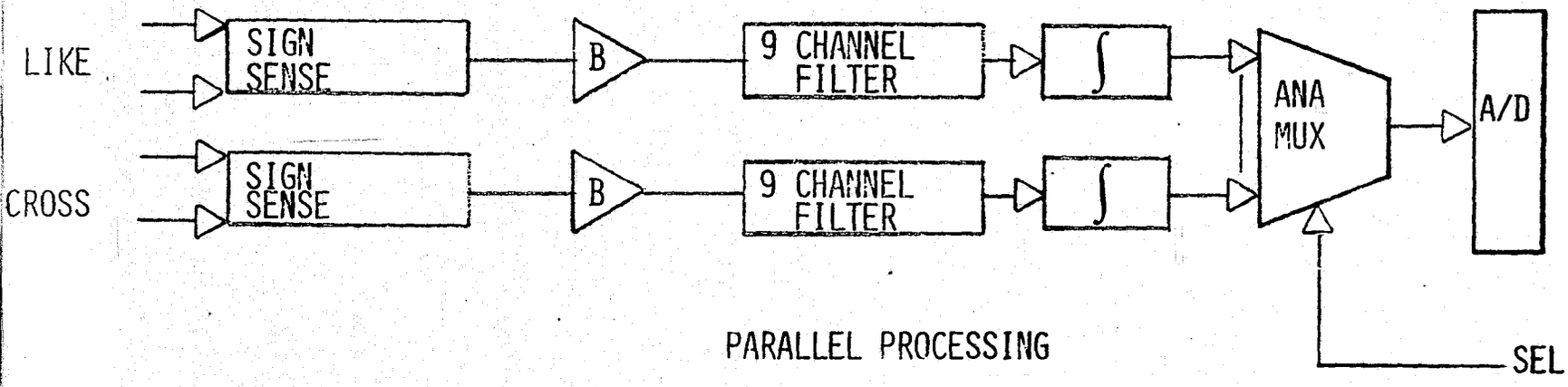
- SIGN SENSE
- POWER SPECTRA SAMPLING
- DECON A/C DATA
- CALCULATE SCATTERING COEFFICIENT
- ALIGN COEFFICIENTS FOR A SINGLE CELL
- OUTPUT DATA TO OPERATOR AND RECORDERS



SYSTEM FUNCTION



SERIAL PROCESSING

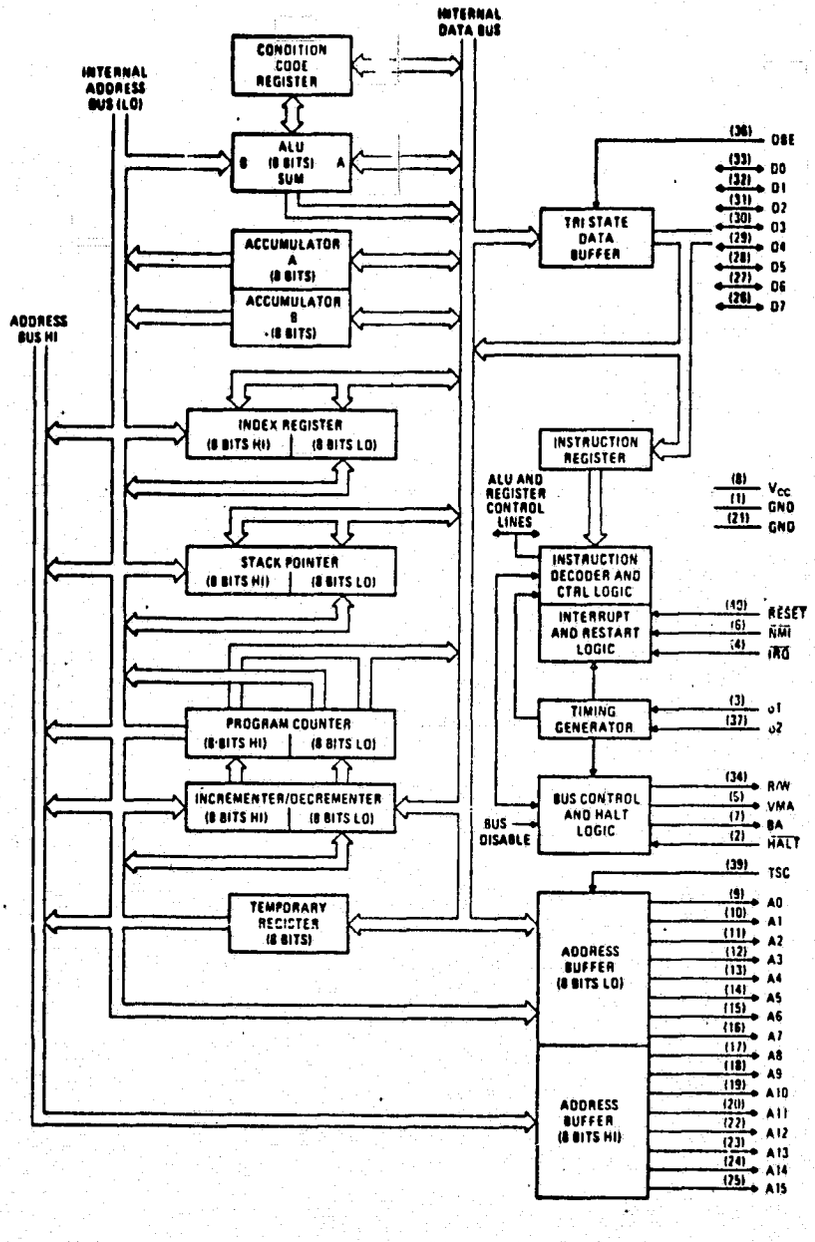


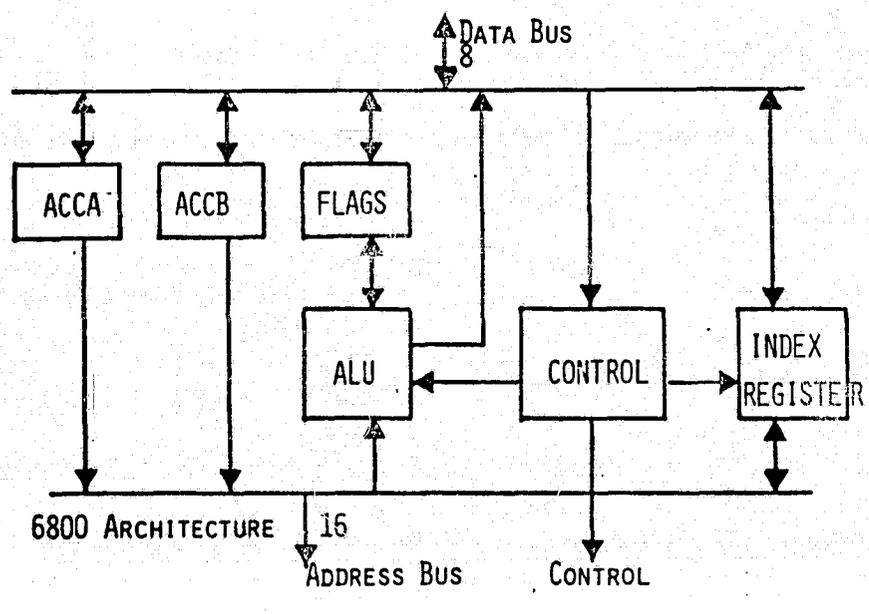
PARALLEL PROCESSING

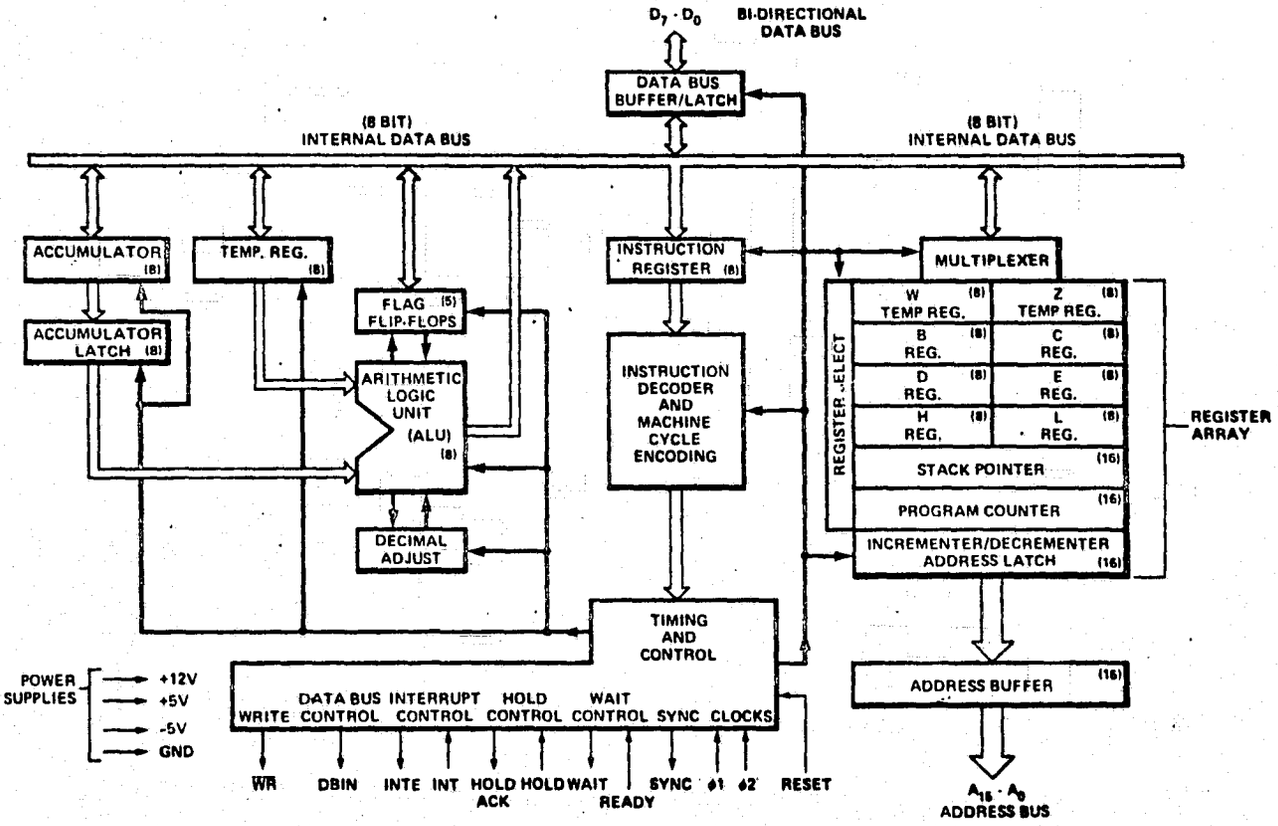
DOPPLER FILTER TRADEOFF

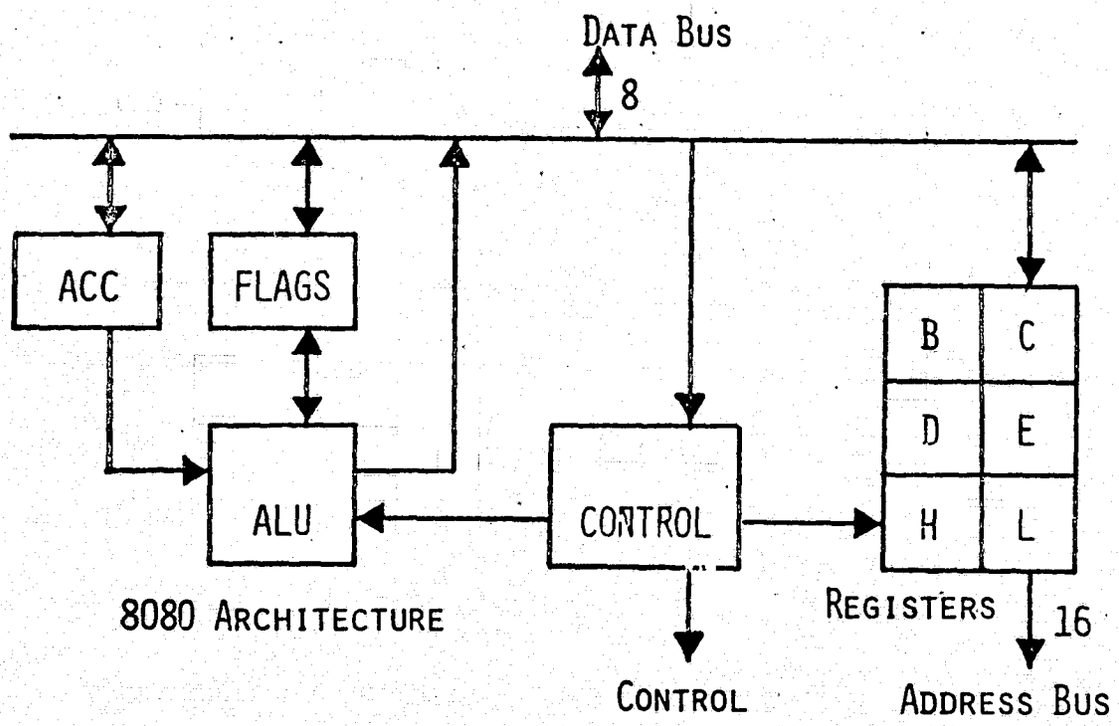
	13.36Hz			4.756Hz			1.66Hz		
	N	L	DB	N	L	DB	N	L	DB
SEQUENTIAL	20	45	1.75	10	61	2.38	5	86	3.21
PARALLEL	200	115	.54	100	130	0.83	50	156	1.15

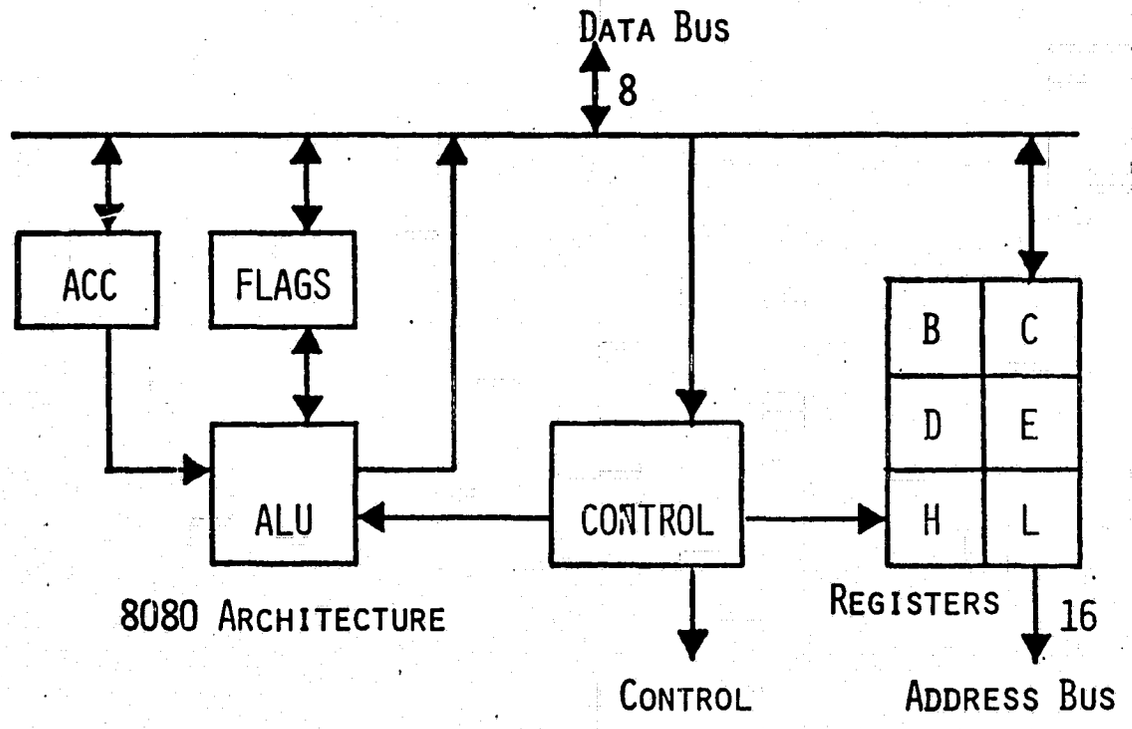
	M6800	INTEL 8080A
INSTRUCTIONS	72	78
8-BIT REGISTERS	2	7
8-BIT ACCUMULATORS	2	1
16-BIT REGISTERS	2	5
INDEX REGISTERS	1	0
ADDRESS MODES	8	7
I/O ADDRESSES	ALL ADDRESSES	256
FLAG BITS	6	5
I/O STRUCTURE	MEMORY MAPPED	ISOLATED OR MEMORY MAPPED
ADDRESS RANGE	64K BYTES	64K BYTES
CLOCKS	1 MHz, 2 ø	2 MHz, 2 ø

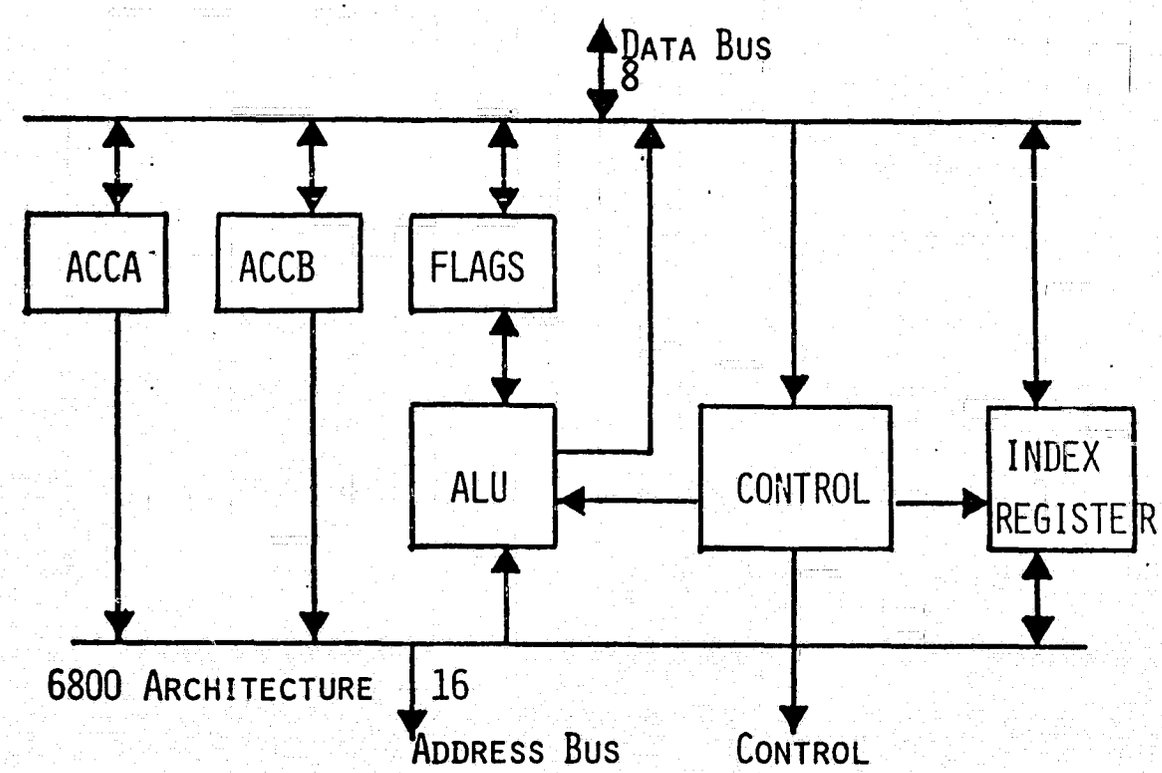

INTERNAL DATA BUS
INTERNAL ADDRESS BUS (LO)
CONDITION CODE REGISTER
ALU (8 BITS) SUM
ACCUMULATOR A (8 BITS)
ACCUMULATOR B (8 BITS)
ADDRESS BUS HI
INDEX REGISTER (8 BITS HI) (8 BITS LO)
STACK POINTER (8 BITS HI) (8 BITS LO)
PROGRAM COUNTER (8 BITS HI) (8 BITS LO)
INCREMENTER/DECREMENTER (8 BITS HI) (8 BITS LO)
TEMPORARY REGISTER (8 BITS)
TRI STATE DATA BUFFER
INSTRUCTION REGISTER
ALU AND REGISTER CONTROL LINES
INSTRUCTION DECODER AND CTRL LOGIC
INTERRUPT AND RESTART LOGIC
TIMING GENERATOR
BUS CONTROL AND HALT LOGIC
BUS DISABLE
ADDRESS BUFFER (8 BITS LO)
ADDRESS BUFFER (8 BITS HI)
(36) DBE
(33) D0
(32) D1
(31) D2
(30) D3
(29) D4
(28) D5
(27) D6
(26) D7
(8) Vcc
(1) GND
(21) GND
(40) RESET
(6) NMI
(4) IRQ
(3) φ1
(37) φ2
(34) R/W
(5) VMA
(7) BA
(2) HALT
(39) TSC
(9) A0
(10) A1
(11) A2
(12) A3
(13) A4
(14) A5
(15) A6
(16) A7
(17) A8
(18) A9
(19) A10
(20) A11
(22) A12
(23) A13
(24) A14
(25) A15


DATA BUS
8
ACCA
ACCB
FLAGS
ALU
CONTROL
INDEX REGISTER
6800 ARCHITECTURE
16
ADDRESS BUS
CONTROL


D7 - D0
BI-DIRECTIONAL DATA BUS
DATA BUS BUFFER/LATCH
(8 BIT) INTERNAL DATA BUS
(8 BIT) INTERNAL DATA BUS
ACCUMULATOR (8)
TEMP. REG. (8)
FLAG (5) FLIP-FLOPS
INSTRUCTION REGISTER (8)
MULTIPLEXER
ACCUMULATOR LATCH (8)
ARITHMETIC LOGIC UNIT (ALU) (8)
INSTRUCTION DECODER AND MACHINE CYCLE ENCODING
W TEMP REG. (8)
Z TEMP REG. (8)
B REG. (8)
C REG. (8)
D REG. (8)
E REG. (8)
H REG. (8)
L REG. (8)
REGISTER SELECT
REGISTER ARRAY
STACK POINTER (16)
PROGRAM COUNTER (16)
INCREMENTER/DECREMENTER ADDRESS LATCH (16)
DECIMAL ADJUST
TIMING AND CONTROL
ADDRESS BUFFER (16)
POWER SUPPLIES
+12V
+5V
-5V
GND
DATA BUS WRITE CONTROL
INTERRUPT CONTROL
HOLD CONTROL
WAIT CONTROL
SYNC
CLOCKS
WR
DBIN
INTE
INT
HOLD ACK
HOLD
WAIT READY
SYNC
φ1
φ2
RESET
A15 - A0
ADDRESS BUS

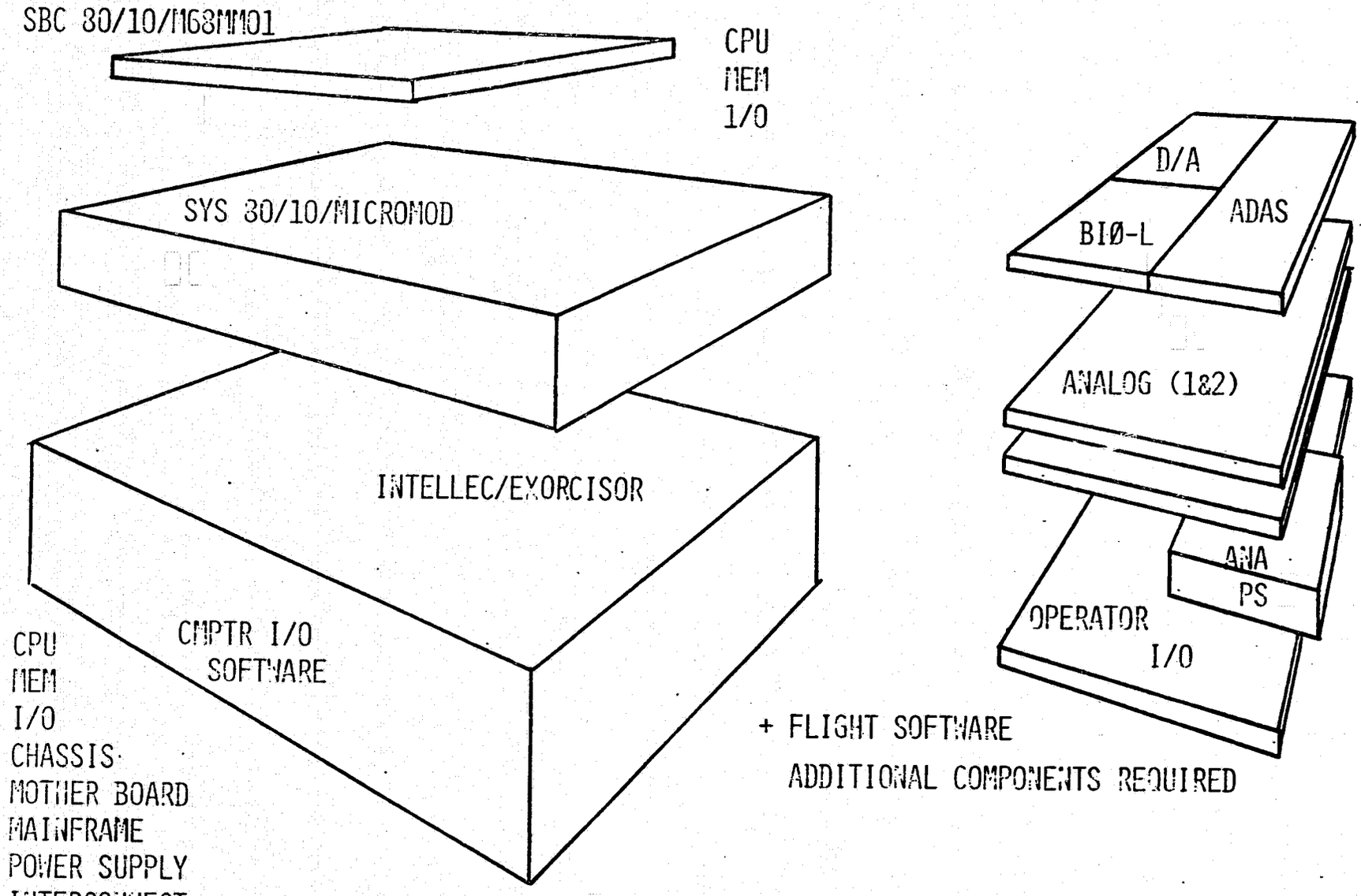

DATA BUS
8
ACC
FLAGS
ALU
CONTROL
B
C
D
E
H
L
8080 ARCHITECTURE
REGISTERS
16
CONTROL
ADDRESS BUS


DATA BUS
8
ACC
FLAGS
B
C
D
E
ALU
CONTROL
H
L
8080 ARCHITECTURE
REGISTERS
16
CONTROL
ADDRESS BUS


DATA BUS
8
ACCA
ACCB
FLAGS
ALU
CONTROL
INDEX
REGISTER
6800 ARCHITECTURE
16
ADDRESS BUS
CONTROL

COMMERCIAL SYSTEM

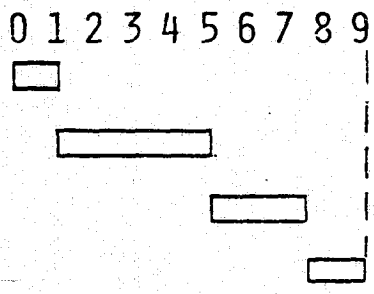
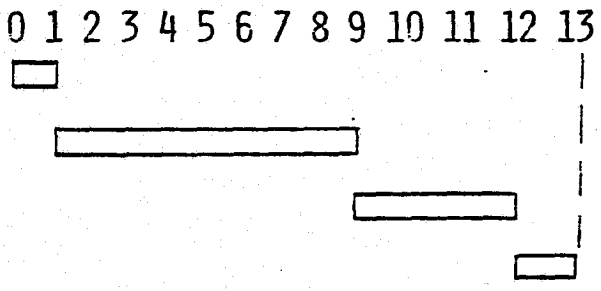
COMPONENTS (+)



+ FLIGHT SOFTWARE

ADDITIONAL COMPONENTS REQUIRED

SYSTEM DEVELOPMENT TRADE-OFF

	WITH COMMON COMMERCIALLY AVAILABLE COMPONENTS PROCURED	ALL FAB AND DEVELOPMENT ACCOMPLISHED IN-HOUSE
SCHEDULE		
COST		
FAB LABOR	X/2	X
SHOP FAB	X/10	Y
MATERIAL	Z	Z
TOTAL	1/2	
DESIGN FLEXIBILITY	—	—
RELIABILITY/QUALITY ASSURANCE	COMMONALITY WITH EXISTING COMMERCIAL COMPONENTS BEST COMMERCIAL PRACTICES	

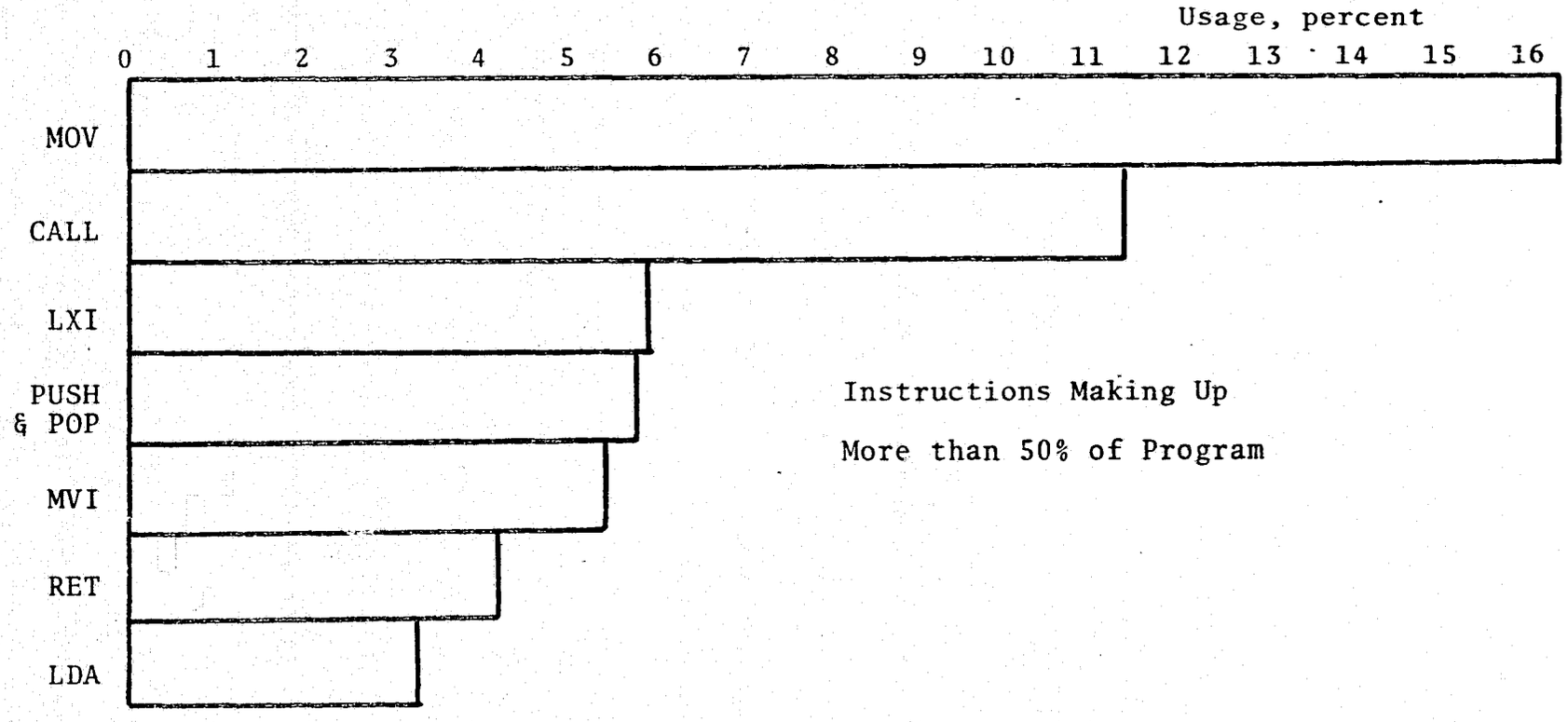
SOFTWARE TRADE-OFF

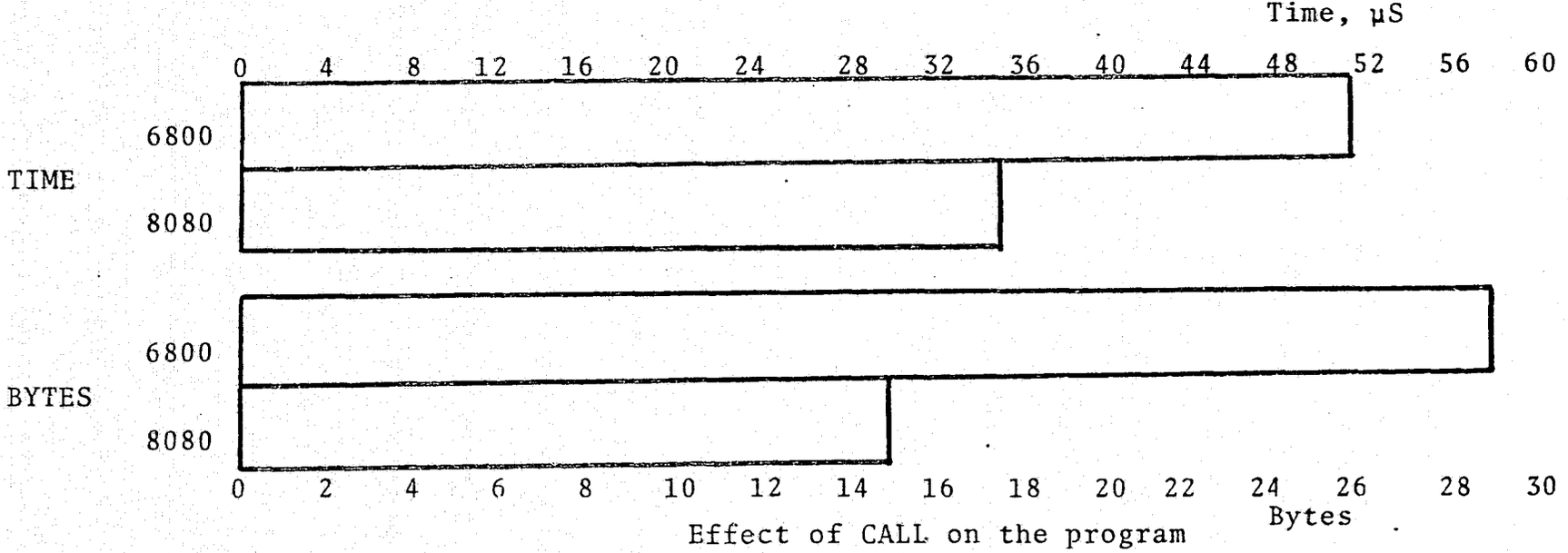
UTILIZING SUBSETS OF EXISTING FLIGHT PROGRAM

TABLE LOOKUP

IN THE 8080 MATHEMATICS SUBROUTINES, THE H REGISTER PAIR IS USED AS AN INDEX INTO THE TABLES, MAKING COMPUTATION OF THE EFFECTIVE TABLE ADDRESS A SINGLE DAD. THE 6800 DOES NOT ALLOW A 16-BIT ADD, SO FOR AN EQUIVALENT OPERATION THE LOOK-UP PROCEDURE MUST BE RECODED.

8080				6800			
BYTES	TIME			BYTES	TIME		
3	5	LXI	H, TAB	3	4	LDAB	TAB+1
2	3.5	MVI	B,0	1	2	ABA	
1	2	ADD	A	3	4	LDAB	TAB
3	5	JNC	NC	2	4	BCC	NC
1	2.5	INR	B	1	2	INCB	
1	2.5 NC	MøV	C,A	3	5 NC	STAA	SAVE3
1	5	DAD	B	3	5	STAB	SAVE3+1
1	3.5	MøV	E,M	3	5	LDX	SAVE3
1	2.5	INX	H	2	5	LDAA	0,X
1	3.5	MoV	D,M	2	5	LDAB	1,X
15	35 µs			3	5	STAA	SAVE3
				3	5	STAB	SAVE3+1
				29	51 µs		





Effect of CALL on the program

CONCLUSIONS

- INTEL 8080 AND MOTOROLA M6800 SERIES MICROPROCESSORS EQUIVALENT FOR PRESENT TASK.
- 8080 SERIES LEADS INTO FURTHER MICROPROCESSOR DEVELOPMENT. 6800 SERIES IS LIMITED WITH RESPECT TO FURTHER DEVELOPMENT.
- 8080 DEVELOPMENT SOFTWARE AND VENDOR SUPPORT EXCELLS SIGNIFICANTLY.
- AVAILABLE MICROPROCESSOR MAINFRAMES ARE ACCEPTABLE AND CAN BE READILY MODIFIED.
- PARALLEL DOPPLER FILTERING IS STRAIGHT FORWARD, LESS FLEXIBLE, MORE EXPENSIVE TO CONSTRUCT AND EASIER TO MAINTAIN THAN SEQUENTED FILTERING.

RECOMMENDATIONS

- AN INTEL 8080 SERIES MICROPROCESSOR BE USED TO IMPLEMENT THE C-BAND PROCESSOR SYSTEM,
- A FIRST LEVEL INTEL MAINFRAME BE PURCHASED FOR THE IMPLEMENTATION, AND THAT ADDITIONAL FUNCTIONS REQUIRED FOR PROCESSING BE ADDED TO THE MAINFRAME,
- A SECOND MAINFRAME CHASSIS BE PURCHASED TO HOUSE THE ANALOG PORTION OF THE PROCESSOR, AND
- THAT A PARALLEL APPROACH TO DOPPLER FILTERING BE USED RATHER THAN THE SEQUENTIAL APPROACH.